ソフトウェア工学 107－19
（1996．1．19）

データフロー図と融合した業務フロー図とそのテンプレートを用いたソフトウェア設計手法の提案

斎　直人
itsuki@slab.ntt.jp
NTT ソフトウェア研究所
〒180 武蔵野市緑町 3 - 9 - 11

ソフトウェア開発の上流工程を改善するために構造化分析手法を導入したが、構造化分析手法による設計書が読めない、あるいは記述できない担当者が多く、導入が進まない。これは、従来の設計方法と構造化分析手法で設計書の記法に違いが大きく、思考の転換が容易でないからである。従来は、機能の理解が容易だが厳密な記法の無い業務フロー図を使用していた。この業務フロー図に、構造化分析手法で使用するデータフロー図の概念を導入・融合した記法と、そのテンプレートを使用したソフトウェア設計手法を考案し、記述実験により有効性を確認した。この手法により、従来の設計方法から構造化分析手法による設計に容易に移行できる。

A software design methodology based on Business Flow Diagram and its templates

Itsuki Naoto

itsuki@slab.ntt.jp
NTT Software Laboratories
3 - 9 - 11 Midori-cho Musashino-shi Tokyo 180 Japan

The structured analysis methodology is difficult for engineers who have original design methodology used by Business Flow Diagram (BFD). They can't write correct documents by BFD, because its description rule isn't clear. This paper describes a softdware design methodology based on New BFD and its templates. New BFD is contrived by unification of usual BFD and DFD. New BFD's rule is clear and useful for software specifications like DFD. Therefore, it is easy to write correct specification by new BFD and templates.

1 はじめに

ソフトウェア開発の上流工程を改善しようという試みが、これまでに数多く行われてきた。特に近年のCASEツールを使用した試みでのねらいは次の点にある。

- 新しい手法の導入による、開発手順の改善と要員のスキルアップ
- 部品化による生産効率の向上
- CASEツールによる、作業の機械化と効率の向上

NTTでも、CASEツールとその基礎となる構造化分析手法[1][2]の導入による、ソフトウェア開発上流工程の改善を進めている。特に要求分析、機能設計作業を対象にして、構造化分析手法を導入し、その図式の一つであるデータフロー図を使用した設計書の作成を行っている。しかし、導入を阻むさまざまな問題が発生し、必ずしも順調に進んでいるとは言えない。その主たる原因は、ソフトウェア開発の現場には長年培ってきた開発形態があり、設計書の構成や記法、そして開発ツールが文化として定着し、新しい形態への早急な転換が難しいことにある。

そこで、従来の開発形態から、構造化分析手法を導入した新しい開発形態に移行する時に発生する問題に対応するために、次のような手段を取った。

- データフロー図と、従来から開発で使用してきた業務フロー図を融合し、新業務フロー図を考案する。
- 新業務フロー図からデータフロー図への変換手順を確立し、構造化分析手法と連結する。
- 新業務フロー図と設計書のテンプレートを用意し、記述効率を向上する。

本論文では、構造化分析手法を導入する時に発生した問題と、その解決策である上記の手段に関する検討と結果を述べる。

2 構造化分析手法導入時の問題

2.1 導入手順

構造化分析手法を、ソフトウェア開発現場へ導入するために、事前の準備と実際の適用という二段階の手順を取った。

事前の準備では、CASEツールと構造化分析手法のやさしい解説書の作成と、適用支援体制の確立を行った。具体的には以下のような項目である。

- ツールは構造化分析手法を支援ツールSoftDAを開発した[3]。SoftDAの使用法を学習するために、通常の操作マニュアルに加えて、初心者向けマニュアルとノウハウ集を作成した。
- 構造化分析手法を学習するためのテキストを作成した。そのテキストには、構造化分析手法の基本事項だけでなく、新たに考案した系統的要求分析手法DREM[4]を入れ、学習を容易にした。
- さらに構造化分析手法の応用的なノウハウ集も作成した。
- ソフトウェア開発現場からの各種問い合わせや問題を解決する目的で、専門家による支援体制を設置した。
- 構造化分析手法とCASEツールの教育を行うカリキュラムと講師の体制を整えた。[5]

これらの準備を整えた上で、実際の導入を行った。その手順は以下のとおりである。

1. 導入プロジェクトを選定した。選定にあたっては、新しい技術に興味があり、時間的余裕があり、構造化分析手法の適用に適したシステムを開発するプロジェクトであるかを基準とした。
2. 導入プロジェクトの開発担当者に構造化分析手法とCASEツールの教育を行った。
3. 導入プロジェクトでの試行を行い、実際の適用形態の検討を行った。
4. 試行結果を評価し、本格的な適用形態を決定し適用した。

これらの手順の中で、2の教育までの過程は支援体制側が主体となって行った。3の試行以降については導入プロジェクトが主体となって行った。なぜなら、試行以降は、開発するシステムに関して詳しくないと、適用形態の検討ができないからである。

2.2 発生した問題

前節で述べたように構造化分析手法の導入を進めて発生した問題を以下に述べる。これらは試行段階で発生し、本格的な適用への進展を阻害する。

(1) 設計書の構成が適切でない。

開発対象システムのモデルを構造化分析手法の図式で表現しても、設計書としてどのように構成するかが明確でない。したがって、開発担当者によって設計書の構成や記述内容が異なり、作業時に戸惑いが出た。

設計書は、構造化分析手法の図式だけで記述できないので、従来からの記述法と合わせて構成しなければならないが、その指針がなかった。そこで、各プロジェクトの開発担当者がそれぞれ従来の設計書構成をほとんど変えずに、構造化分析手法の図式を取り入れた設計書を作成した。

このため設計書の構成がプロジェクトにより異なり、そこに記述するデータフロー図の位置付けもさまざまになり、設計結果を適切に表現できなかった。例えば、構造化分析手法では物理モデル、論理モデルという区別をするが、実際の記述ではそれらのモデルの区別があいまいで混合したデータフロー図になった。さらに、階層的に記述するデータフロー図の、各レベルでの内容、記述する階層の深さも統一されなかった。

(2) データフロー図が読めない。

システムの開発担当者は構造化分析手法を学習したのでデータフロー図等を理解できるが、構造化分析手法を学習していない開発依頼者は読めなかった。

導入プロジェクトでのシステム開発は、開発担当とは別の部所からの開発依頼で行っていた。そのため要求分析や機能設計の完了時に、開発担当者と開発依頼者が共同で設計書のレビュを実施した。しかし、構造化分析手法を知らない開発依頼者にとって、データフロー図等の図式は理解できない記述であった。このため設計書のレビュ時には、開発担当者がデータフロー図等の読み方を説明するために時間を割くことになり、データフロー図を理解して行う実質的なレビュができなかった。

データフロー図が読めないとなると、設計書への記述は取りやめとなり、単なる設計途中のメモとして使用することになる。したがって、構造化分析手法導入の効果が十分に出ない。。

(3) データフロー図が書けない。

データフロー図で自由にシステムを表現するのは容易ではない。

データフロー図は、プロセス、データストア、外部エンティティとデータフローだけから成る図式であり、システムを抽象化して階層的に記述する。その記法を学習し、簡単なデータフロー図を読み取れるようになるのは容易である。しかし記法を拾得するだけでは複雑なシステムのデータフロー図を読み取ったり、記述することはできない。

データフロー図が容易に記述できない主な理由は以下のとおり。

・実際のシステムの機能を連想できない。

使用する記号の種類が少なく、抽象化しているのが特徴であり、初心者には機能を理解することが難しい。例えば、データの蓄積場所を示すデータストアが、実際には電子的なデータベースであるのか、物理的な帳簿であるのか区別しないので、実現する業務と関連づけてデータフロー図を理解することが困難である。

・処理の順序が表現できない。

データフロー図は処理の順序を明に表現しない。これは、フローチャートのような手順を示す図式を用いてシステムを表現してきた開発担当者にとって、発想の転換である。しかし転換が容易にできず、データフロー図の記述を困難にする。

・システム全体を瞬時に把握できない。

構造化分析手法では、詳細な機能は階層的に下位のデータフロー図に記述し上位では隠ぺいする。したがって、一枚のデータフロー図で全体から詳細までのすべての機能を見渡すことができない。従来の設計では大きな用紙一枚にシステム全体を詳細まで記述することが多く、そのような方法になれた開発担当者は、データフロー図の階層化した表現では全体を把握できないと不安を抱く。

・従来より深い検討が必要である。

データフロー図には、あいまいな表現を許さない記述規則が決められている。例えば、意味の異なるデータには、すべて異なる名称を付ける。これを厳密に適用すると、非常に多くの名称が必要になり、名称が長くなったり、命名が困難になる場合がある。従来は、そのような詳細な識別をしない（すなわち、検討が不十分な可能性を含む）設計のため、構造化分析手法を適用することで、検討時間が長くなる。

3 問題解決の方向

これまでに述べたように、構造化分析手法を導入することで、いくつかの問題が発生した。それらは、開発担当者が構造化分析手法の導入を中止するに十分な理由であった。

しかし、従来の開発形態のままでは改善が進まない。

従来の設計書では図式の記法が厳密でなく、開発担当者による差異があり、あいまいである。従来の設計書にはさまざまな図、表、そして文章が、設計

情報項目別に記述されている。それらの記述項目から、データフロー図に最も近い図1のような業務の処理を表現した業務フロー図である。業務フロー図は、要求定義・機能設計工程においてシステム全体の概略を定義、理解するために重要な図となっている。しかし業務フロー図は厳密な記法を持たず、あいまいなくシステムを定義できない。このため、システムの品質低下の一因になっている。

業務フロー図の記法のあいまいな点は次章で述べる。

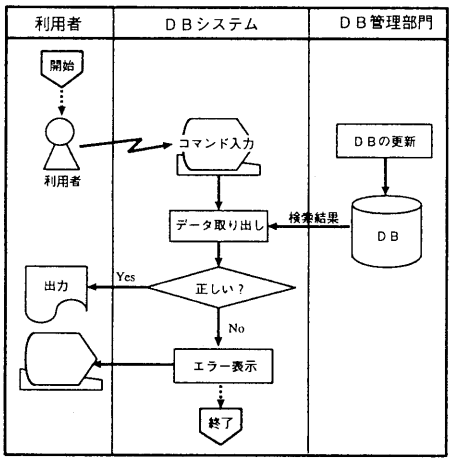


図1　業務処理フロー図の例

これまでの構造化分析手法の導入では、設計書の構成を変えず、データフロー図等の図式を有効に使用できないため、ソフトウェア開発の改善は順調に進まなかった。一方、従来の開発形態は改善が必要である。そこで、次のような手段により開発形態を変え、構造化分析手法の適用に導くこととした。

- 従来のソフトウェア開発で使用してきた業務フロー図とデータフロー図の記法を融合し、新業務フロー図を考案する。
- 新業務フロー図からデータフロー図への変換手順を確立し、構造化分析手法と連結する。
- 設計書と新業務フロー図のテンプレートを用意し、それらを使用して記述効率を向上する。

次章以降では、これらの手段について詳説する。

4　データフロー図と業務フロー図の融合

4.1　業務フロー図記法の分析

データフロー図で構造化分析を行うと、機能を階層的に整理し、データとの関係を明確に定義できる。これは、以下のようなデータフロー図の特徴、規則に依るものである。

- 記述対象をプロセス、データフロー、データストア、外部エンティティ、の4種に分類、抽象化する。使用する記号はそれら4種に限る。
- 記号間はデータフローで接続する。その接続規則が定められている。
- すべてのデータフロー、記号に明確な名前を付ける。
- 相互のデータフローが論理的に矛盾しないように記述する。

そこで、従来設計書の業務フロー図を、データフロー図の特徴、規則と比較し、記法としてあいまいな点を分析、抽出した。その結果、以下のことがわかった。

- 記号の種類が多い。また、同一の記号をデータフロー図のプロセス的に使用したり、データ的に使用したりして、記号に厳密な意味付けをしていない。
- 記号間の接続がデータフローであったり、制御信号であったりして、意味付けが明確でない。また、その接続規則がなく、任意の記号間に接続できる。
- 名前を付けていない記号や接続がある。付いている名前は、同一でも意味が異なることがあり、厳密性に欠ける。
- 記述対象システムと、その他のシステムの区別を表で行っているが、対象システムとその他の区別があいまいである。

4.2　新業務フロー図の記述規則

これまでに述べた問題を解決し、構造化分析手法の導入を進めるために、導入が容易な新業務フロー図を考案した。その記法は、業務フロー図における具体的な機能の連想が容易な性質と、データフロー図における仕様の表現が厳密に出来る性質を融合して、

- 業務フロー図で使用している具体的なイメージを与える記号を使用する。
- データフロー図で定めている明確な記法、特に処理とデータを厳密に区別した記号と接続規則を取り入れる。

とした。

新業務フロー図の記法は次のような手順で定めた。

1. 従来の業務フロー図で使用している記号とその用途、接続規則を抽出する。
2. 業務フロー図から抽出した記号を、データフロー図のプロセス、データストア、外部エンティティに対応させ、意味付けを明確にする。
3. データフロー図のデータフローに相当する記号間の接続規則を流用して、各記号間の接続規則を決める。ただし、データフロー図では記述可能な接続でも、記号の意味を考慮してデータの流れが適切でない場合は、接続を認めない規則とする。

以上の手順により定めた主な規則は次のとおり。

- 各記号を分類し、それぞれの記号グループをデータフロー図と同様に、プロセス、データストア、外部エンティティと総称する。
- 記号間を接続するフローは、方向を持つ矢印付き実線とし、矢の向きがデータの流れを示す。フローには必ずデータ名を記する。
- 業務フロー図で使用している外枠の使用は、外部エンティティを明確にする目的として使用できる。

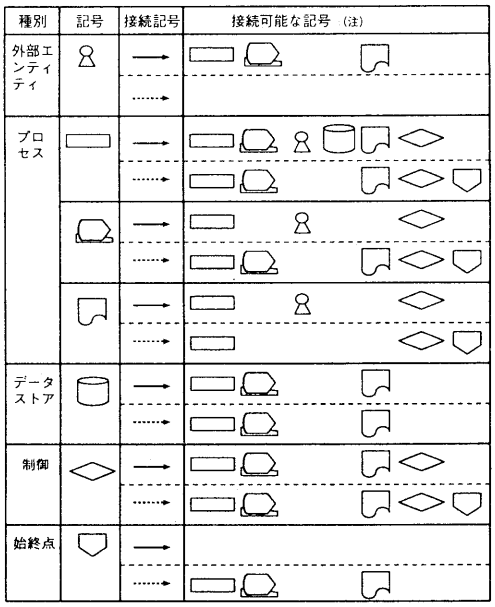


図2　新業務処理フロー図の記号使用規則

(注)「対応する記号」から「接続可能な記号」に向けたデータの流れを矢印付きの実線で表記できる。

4.3　記述実験

前章で述べた新業務フロー図の規則に従って、開発担当で記述試験を行い、適用性を調べた。実験は、2.2で述べた問題が発生したプロジェクトの一つで、記述規則に従って従来の業務フロー図を開発担当者が書き換え、レビュすることで行った。

その結果、従来の業務フロー図と同様の理解性があり、しかも仕様を厳密に記述できるようになることが確認できた。しかし、従来は使用していたが、規則で規定していない条件判断（菱形記号）や制御信号の流れを表現したいという要望があった。それらはデータフロー図で表現しないシステムの制御を示すため、当初の記述規則では省いていた。しかし実用的に必要なため、データフローに相当する接続の途中で使用する出来るように拡張した。また、制御信号の流れは破線で表現できることとした。

記述実験結果により変更し決まった記述規則の一部を図2に示す。

記述規則に従って図1の業務フロー図を書き変えた新業務フロー図を図3に示す。

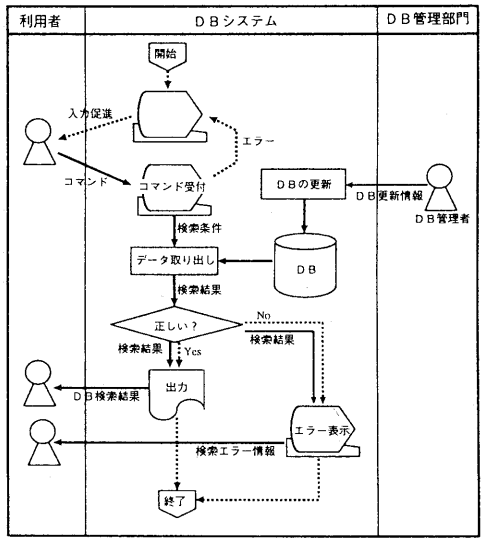


図3　書き換えた新業務処理フロー図

4.4　データフロー図への変換

新業務フロー図はデータフロー図に近い記述だが、データフロー図ではない。記号の種別や接続規則はデータフロー図に準じており、仕様を明確に記述で

きる。しかし、記号が異なり、外枠があり、制御を示す記述が残っている。

最終的には構造化分析手法の適用が目的である。新業務フロー図による完全な構造化分析手法の適用は不可能であり、そのまま構造化分析手法に従ったCASEツールの利用もできない。

そこで、新業務フロー図をデータフロー図に変換する必要がある。この変換は、記述規則が類似しているので、機械的に容易にできる。以下で、図3の新業務フロー図をデータフロー図に変換する例を用いて、変換方法を示す。

(1) 制御記号を消す。

データフロー図では制御を表現しないので、制御記号を消す。このとき、制御先が複数のプロセスであれば、それらを一つのプロセスにまとめ、データフローはその新しいプロセスに接続する。まとめられた元のプロセスの部分は、下位のデータフロー図とする。

図3の新業務フロー図から制御記号を消すと図4のようになる。また、下位のデータフロー図は最終的に図5のようになる。

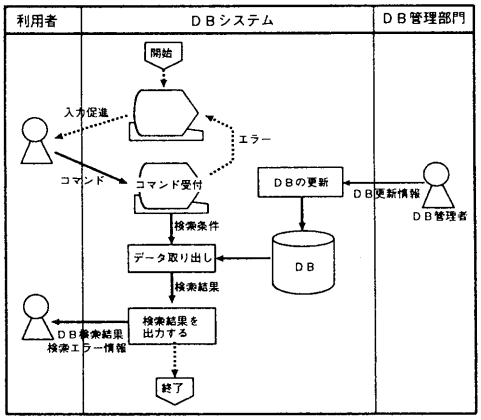


図4　制御記号を消す

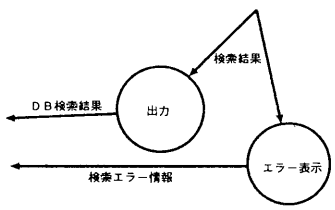


図5　下位のデータフロー図

(2) 制御フローを消す。

制御フローを消す。このとき、制御フローだけが接続しているプロセス記号があれば同時に消し、機能を制御フローの発出元のプロセス記号の機能に統合する。機能を統合すると名称を適当に変更する。また、同時に始終点記号も消す。

図4の例で、制御フローを消すと図6のようになる。

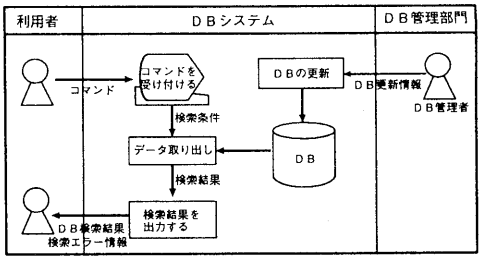


図6　制御フローを消す

(3) 記号を置き換える。

プロセス、データストア、外部エンティティの各記号を、データフロー図で使用する記号に置き換える。同時に外枠を消す。

図6の各記号を置き換えると図7のようになる。

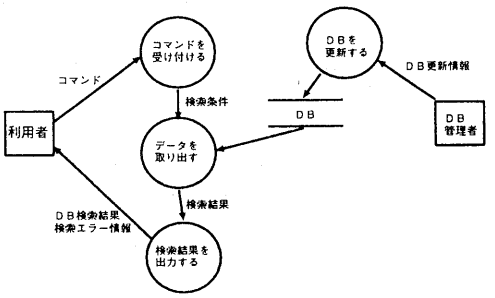


図7　各記号を置き換える

(4) データフロー図を完成させる。

変換したデータフロー図をチェックし、不足する機能やデータを補ったり、データをまとめ、データフロー図を完成させる。この段階はすでに、構造化分析そのものである。

5　テンプレート

5.1　テンプレートの構成

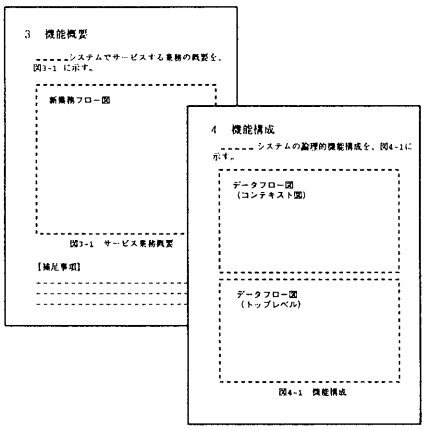
テンプレートの目的は、設計書の記述を適切な方向に導くことにある。構造化分析手法の導入によっ

て発生した問題は、設計書の構成とデータフロー図等の記述にあった。そこで構造化分析手法を導入した開発に適合した、設計書構成と図式の指針としてテンプレートを用意する。すなわち設計書構成用テンプレートと図式記述用テンプレートの2種になる。

(1) 設計書構成用テンプレート

設計書の構成を規定するテンプレートである。単なる目次ではなく、各項目内で使用する図式まで規定する。それによって、新業務フロー図やデータフロー図の記述箇所が明確になる。記述箇所が明確になると、そこで図式に記述する内容レベルや、設計時期が規定されることになる。

図8に設計書構成用テンプレートの例を示す。テンプレートは、全体の構成と、図式を書き込む場所、補足等の図式以外の表現を書き込む場所等を定めている。



波線部がテンプレートの書き込み場所である。

図8 設計書構成テンプレートの例

(2) 図式記述用テンプレート

図式記述用テンプレートは、新業務フロー図等を記述するための部品群である。それらを組み合わせて、図式を記述する。実際には、適当なテンプレートを組合せ、プロセスやデータの名称を確定し、データフロー等を補って完成させる。

図9に、新業務フロー図のテンプレートの例を示す。

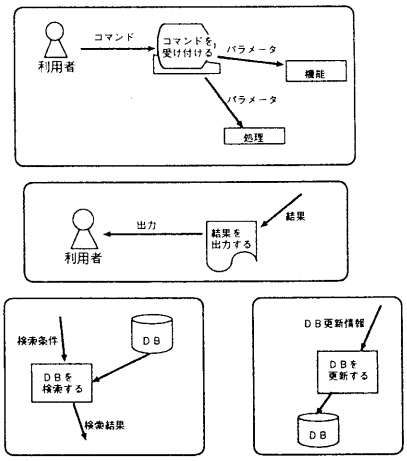


図9 業務フロー図のテンプレートの例

5.2 設計書の記述実験

テンプレートを使用した設計書の記述実験を行った。実験は、既存の設計書の一部を新業務フロー図を使用した設計書に書き換え、テンプレートを抽出し、そのテンプレートを使用して別のデータベースを扱うシステム（例1）と扱わないシステム（例2）の機能設計書を記述し、テンプレートで記述できる範囲を測定した。記述範囲の測定は、図式に記述した記号の中で、テンプレートを使用して記述した記号の数を比率を求める方法で行った。

設計書構成用テンプレートは1種で、そのまま使用した。図式記述用テンプレートは、12種（データベースアクセス用5種、その他7種）を抽出した。

図式記述用テンプレートの使用率を測定した結果を表1に示す。プロセスの6割、データフローの4割をテンプレートで記述できた。

データフローの率が低いのは、テンプレート相互を接続するデータフローを、テンプレートを展開した後に記述する必要があったからである。

テンプレート使用率は、高いとは言えないが、記述の半数をテンプレートで記述することで、その後の記述の見通しができるので、その効果は数値以上に高い。

表1 テンプレート使用率

種別	例1	例2
記述枚数	6枚	35枚
プロセス	72％	55％
データストア	100％	—
データフロー	37％	39％
全記号	50％	46％

6 まとめ

これまでに述べた手段によって、ソフトウェア開発に構造化分析手法を導入することが容易になる。それは次の手順で開発を進めることができるからである。

1. 設計書構成テンプレートに従って、設計書の構成、設計手順を確認する。
2. 業務概要を新業務フロー図で記述する。この時、図式記述用テンプレートを使用する。
3. 変換手順に従って、新業務フロー図をデータフロー図に変換し、機能構成を記述する。
4. データフロー図が完成するので、以降は構造化分析手法に従って開発を進める。

2.2 で述べたように、導入初期に開発担当者は物理モデルと論理モデルの区別ができなかった。構造化分析手法ではどちらもデータフロー図で記述し、その記法的な違いはないからである。しかし、上記の手順のように、まず新業務フロー図を記述し、それを変換してデータフロー図を記述することで、新業務フロー図が物理モデル、データフロー図が論理モデルとなり、区別が明確になる。したがって、新業務フロー図からデータフロー図への変換手順は、物理モデルから論理モデルへの変換手順でもある。

7 おわりに

本論文では、新業務フロー図とテンプレートによる設計手法を提案した。また、記述実験によって、新業務フロー図の理解性と、テンプレートの有効性が確認できた。今後は、さらにソフトウェア開発への適用を進め、手法の充実を図る予定である。

[謝辞] 本検討の機会を与えて頂いたNTTソフトウェア研究所黒田幸明主幹研究員、協力して頂いた各位に感謝する。

[参考文献]

[1] Edward Yourdon： Managing the Structured Techniques、1985（邦訳、黒田純一郎、渡部研一訳：構造化手法によるソフトウェア開発、日経マグロウヒル社、1987）

[2] Tom DeMarco： Structuerd Analysis and System Specification、1979（邦訳、高梨智弘、黒田純一郎訳：構造化分析とシステム仕様、日経マグロウヒル社、1986）

[3] 磯田定宏、黒木宏明：統合化CASEシステムSoftDAの機能 — 上流と下流の統合化、コンピュータソフトウェア、Vol.10、No.28(1993)、PP.26-37

[4] 山本修一郎、斎 直人：系統的要求分析手法DREM、ソフトウェア工学の基礎ワークショップ94、日本ソフトウェア科学会、1994

[5] 斎 直人、田中 清、山本修一郎：構造化分析設計技法の教育と適用、利用者指向の情報システムシンポジウム、情報処理学会、1993